SUPPORTING THE MODELING LIFE CYCLE

S.C. BANKES,* Evolving Logic, Los Angeles, CA

ABSTRACT

Computational science has for the most part been focused on creating appropriate representations of problem domains of interest and reporting the results of modeling experiments based on those representations. This focus has caused other equally important issues to be neglected. In order for computational science to achieve its potential, methods and technologies are needed to support the entire modeling life cycle, including model creation, modification, and exploitation. In this paper, I discuss these challenges and provide examples where initial steps have been taken to meet them.

Keywords: Computational science, model exploitation, exploratory modeling, computer-assisted reasoning, robust inference

INTRODUCTION

The idea of support throughout a life cycle has a substantial history. Our current understanding has grown out of a natural but naïve concept of life cycle as a linear series of stages. For nearly any artifact, it seems natural to begin with the concept for the product and proceed to analysis, design, and implementation, arriving finally at a functional system. That system will get used and maintained and finally be retired.

Such a linear process, while intuitively comfortable, suppresses the iterative nature of the design and construction process. And as an ideal, it can lead to many disasters, as it implies a process that contacts the world only at beginning and end. Often, this can result in a product that is rejected by the user when it is fielded, even though users were fully consulted during the specification and design phases.

A notorious version of the linear approach is the waterfall development methodology, which has been used for software engineering and has been blamed for many software engineering failures. Although it is still in use here and there, the waterfall methodology is identified in many texts on software engineering as a discredited approach. Certainly, it cannot be recommended for programming in general, as experience has shown it to lead to systems that do not track shifting requirements and that can be difficult to maintain. There are a variety of methods that are preferred, but a prominent example designed to counter the natural tendency towards a waterfall approach was the spiral development method (Boehm 1988 and 2000).

Spiral development encourages us to consider a cyclic process in which specification, design, implementation, and testing occur repeatedly, with new features appearing and design errors being fixed with each cycle. Related ideas such as rapid prototyping also encourage engagement with the user throughout the development process. It is an implication of such an

* *Corresponding author address*: Steven C. Bankes, Evolving Logic, Inc., 2118 Wilshire Blvd., #423, Santa Monica, CA 90403; e-mail: bankes@evolvinglogic.com.

approach that development is never truly completed, and that the distinction between development and maintenance is not hard and fast.

Computational science has its own version of the waterfall process, and just as with the early years of software engineering, it is viewed as so obvious, it is seldom questioned. Roughly, the stages of a linear process of computer modeling are:

1. Gather all relevant data and theory.

2. Design the model.

3. Implement the model, including debugging and verification.

4. Load the inputs corresponding to some question.

5. Run the model once or a series of times in order to answer the question. If it is a policy question, search for the policy that produces the best model outcomes.

The waterfall approach to computational science has led to many problems, including both specious research and a blindness to opportunities for rigorous research that are overlooked in the search for predictive accuracy. This leads one to ask the question, "What is the equivalent to spiral development for computational science?" A different way to express this need is to say that we must move beyond the focus on models, to provide better support for the activities of modeling. Models by themselves provide no value. The value arises in the context of a web of relationships that must be supported and utilized. These include:

- *Beliefs that motivate and explain the structure of the model.* Of special note are assumptions built into the model that must be noted if inference based on the model is to be rigorous.

- *The experts who are the source of these beliefs and assumptions.* They will often be the model's authors, but could include others. Since these experts may have beliefs and assumptions that they implicitly hold, this meta-information is also important to rigor in model-based science.

- *The data used in model construction.* These data inform the model just as much as beliefs do, so its provenance is should also be captured. Increasingly, model data will be updated regularly, perhaps with real-time feeds. Technology to make this easy, to provide configuration management and to verify the process, is needed as part of the infrastructure for modeling.

- *The relationship of a model to its users.*

- *The relationship of a given model to alternative models as well as to previous versions.* An audit trail or change history is one example of how this relationship can be made explicit and supported. A deeper need is to represent the differences between models in terms of the assumptions they embody, so that joint inference can be supported.

- *The relationship to other, complementary models.* Great benefits could be achieved if models of different phenomenology could be used together to understand the union of the information they embody. Simply gluing the models together may not be the best means for achieving this, however.

- *Data from cases run on the model.*

- *Analytic materials derived from the model.* Examples are statistical analyses or graphical visualizations.

- *Arguments and conclusions based on the model.*

- *Other software systems and applications to which the model may be linked.* Examples are decision support tools or control systems.

SOFTWARE TO SUPPORT COMPUTATIONAL SCIENCE

An example of a system that provides at least some categories of support for "modeling" is the computer assisted reasoning system (CARs™). The original design for CARs was inspired by the concept of exploratory modeling (Bankes 1993). That is to say, the design makes computational experiment the central concept, and provides various facilities for using models to conduct computational experiments, and for using the results of computational experiments to support reasoning. CARs' design emphasizes providing options for interactive use, and for managing large ensembles of models and cases. It has been used as a basis for implementing methods for robust decision analysis, and has been applied to a variety of planning and decision problems. In the discussion below, examples from the design for CARs and its use on various projects will be provided.

CARs is a Java-based system that links to virtually any type of model, treating it as a "scenario generator." In particular, CARs can be harnessed with models implemented in C++ or Java, or any language that supports sockets or Micosoft's COM. CARs provides a variety of services that can be applied to any model once it has been harnessed, including interactive use of search and visualization to create, explore, compare, and understand very large ensembles of scenarios with ease.

In CARs, a scenario generator (aka model) is encapsulated in an object that augments its input/output behavior with a variety of other services, including databases of cases (input/output pairs) and methods that help other parts of the system customize their behavior appropriately. Software clients for these services can interact with objects of this class (known inside of CARs as a "context") transparent to the means that are being used to run cases. Aside from response time, other objects are free to regard contexts as repositories of every possible case that could be run, virtually representing ensembles of what-if questions that can be of infinite cardinality or infinite dimensionality. Keeping a database of previous runs can improve response time, but does not fundamentally alter the logical requirements for making inferences about an infinite set from a finite sample.

On the basis of software architectures such as that of CARs, a variety of avante garde methodologies can be entertained. For some applications, these methodologies may be more

appropriate than a realist strategy of model use that has been so successful for engineering models used for computer assisted design. For example, Figure 1 is taken from work in which a coevolutionary strategy has been employed for robust inference. Here virtual ensembles of plausible future challenges are allowed to evolve in parallel to a virtual space of possible coping strategies. Sampling out of each of these sets is guided by the other, seeking assumptions about the future that are maximally stressing for the leading candidate strategies and strategies that perform well across the challenge landscape.

THE CHALLENGE OF MODEL EXPLOITATION

Computational science can be seen as a sequential (and iterative) process of divergent and convergent thinking. Instead of having only a single image of the future in mind (left side of Figure 2), the idea is to confront uncertainty and recognize as much as possible of the highly diverse futures that are in fact plausible. There are two universes of cases in the center of the figure. The larger one is the virtual ensemble of all cases that might possibly be run, while the smaller is the corpus of cases that actually have been run.

Just generating cases is not enough; the next challenge is to make use of the results — to apply more convergent methods for sense making and for the discovery of insights. Various techniques can be used to accomplish this, including interactive visualization, search techniques to run cases that are most likely to be salient to the problem at hand, and the use of data mining algorithms to extract information from the resulting database of cases. (Also interesting is the hybrid set of algorithms called active learning or adaptive sampling that combine search and data mining properties.)

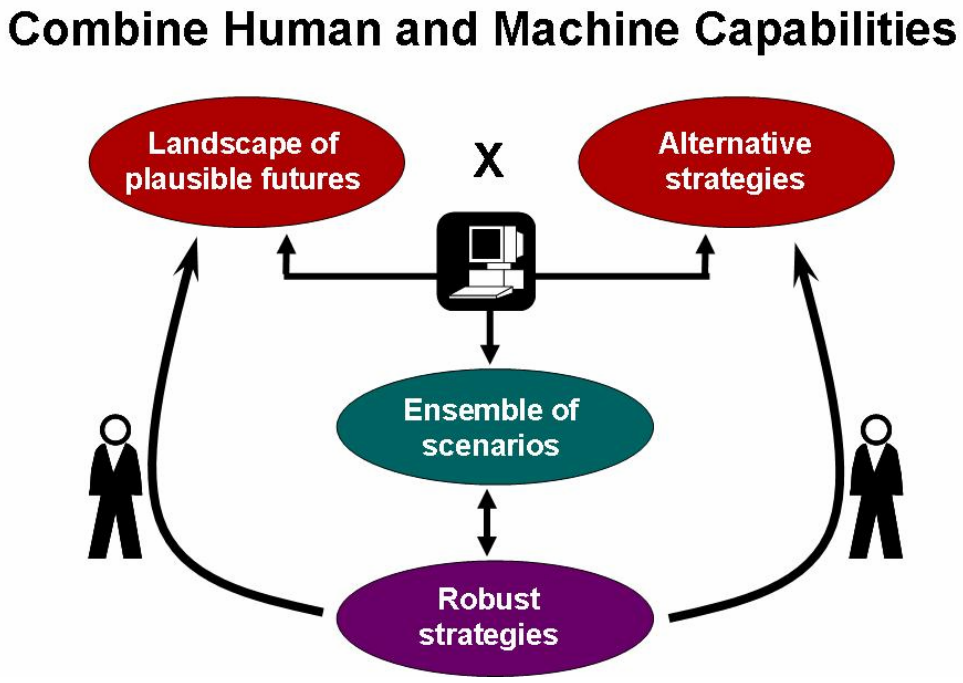

FIGURE 1 Coevolutionary analytic methodology

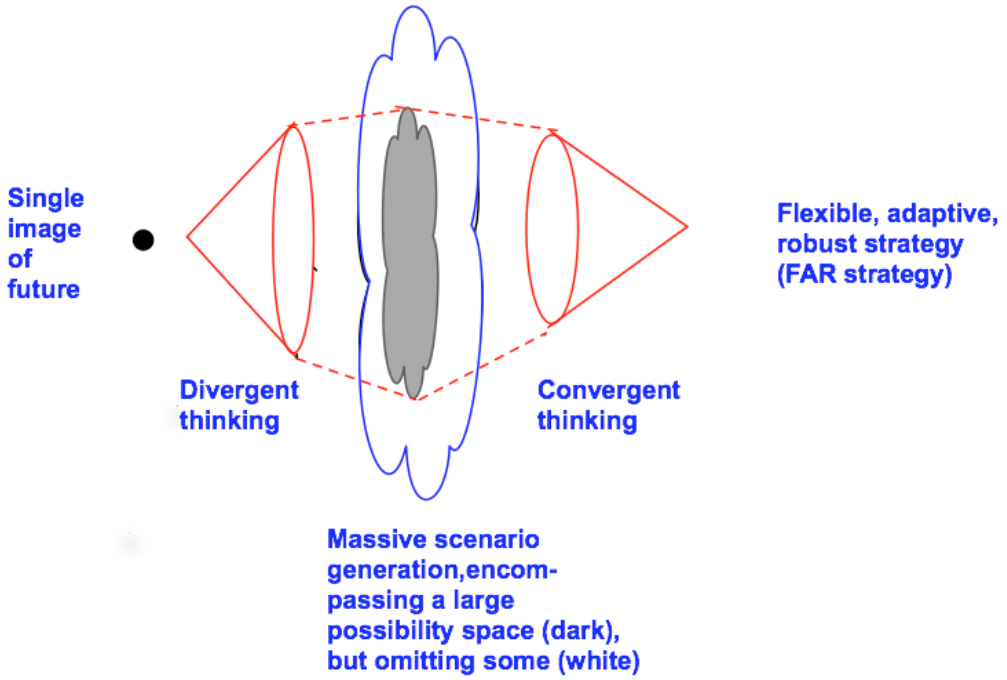

FIGURE 2 Diverging and converging

This approach suggests a series of implications regarding how best to contend with difficulties that have confounded computational science in the past, and opportunities to improve its impact on the future. In the remainder of this paper, I give a few examples that I and colleagues have begun to address.

CONFRONTING THE CURSE OF DIMENSIONALITY

The first of these challenges is the so-called "curse of dimensionality." In brief, the number of cases required to assess the response of a model rises geometrically with the dimensionality of the input space. In a rectangular region of an N-dimensional input space, assessing the range of response requires 2^N cases if the response is monotonic throughout the region. If the response surface is more rugged, no upper bound on the number of cases exists, and even for the monotonic case, 2^N cases can become impractical for a relatively modest N.

Sensitivity analysis as typically practiced does not attempt 2^N cases, but merely 2^N excursions around a baseline case. This approach assumes a linear model. Its application to nonlinear models amounts to an asymptotic argument of linearization for small enough excursions, meaning that one must assume that the uncertainty range in the inputs is quite small. For nonlinear models with significant uncertainty in the inputs, making a best estimate prediction with sensitivity analysis is not a coherent research strategy. It has been the only strategy available to researchers, so it has been used, with or without apologies. The curse of dimensionality is typically only mentioned in the context of providing an excuse for leaving out the sensitivity analysis part of this method.

Robust inference provides a superior basis for reasoning based on computational experiments on problems with deep uncertainty (Bankes and Lempert 2004). Robust inference is based on identifying a class of experiments and a proposition such that the proposition has been true for all experiments conducted that satisfy the class definitions. The strength of such an inference depends both on how many cases have been examined and how cleverly they were crafted to try to invalidate the proposition. All of the following discussion is based on this idea, and the following provides a few examples of its use.

REASONING FROM MULTIPLE MODELS

Often, there is not a single parameterized model structure that is sufficient to express all of the possibilities consistent with our knowledge. The ability to reason across model structure is useful in a variety of contexts. Figures 3 and 4 exhibit two different Bayes nets, built from expert elicitation, that capture two different hypotheses. In this case, the two hypotheses have to do with the nature of a terrorist threat, and these two diagrams capture for a notional counter-terrorism problem the difference between a threat that is organized hierarchically and one that is more networked.

In order to make robust inferences across this sort of model resident knowledge, it is necessary to find conclusions (about policies in this case) that are robust to uncertainties about both the structure of the model and the parameters that characterize any particular model instance.

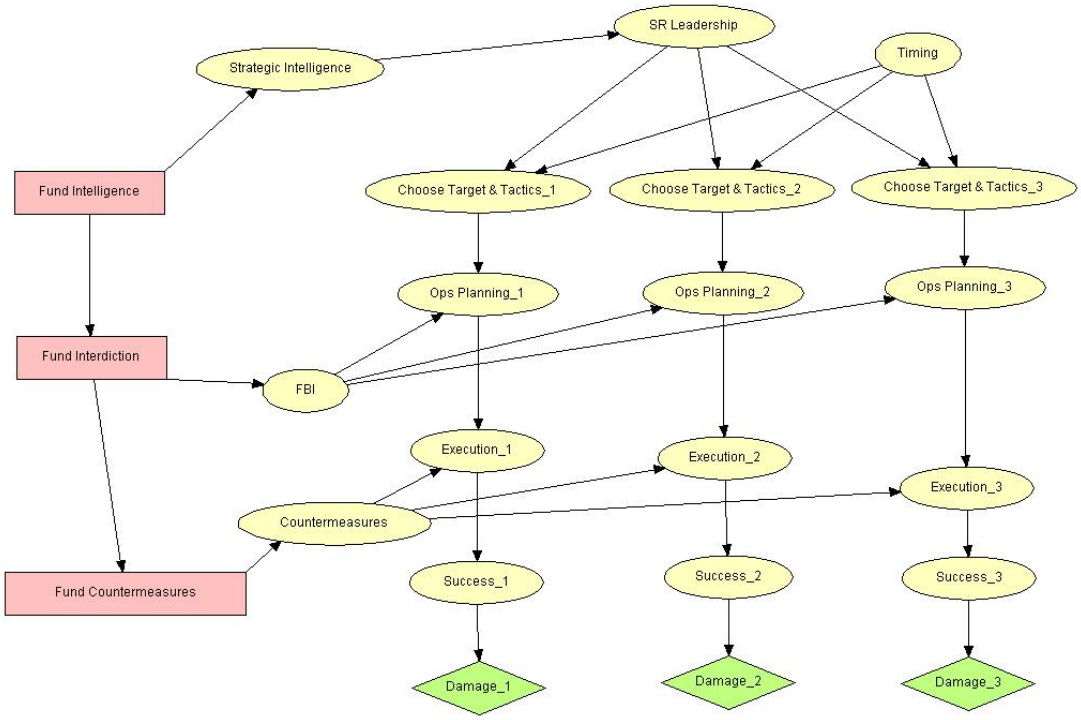

FIGURE 3 Counter-terrorist model for network threat organization

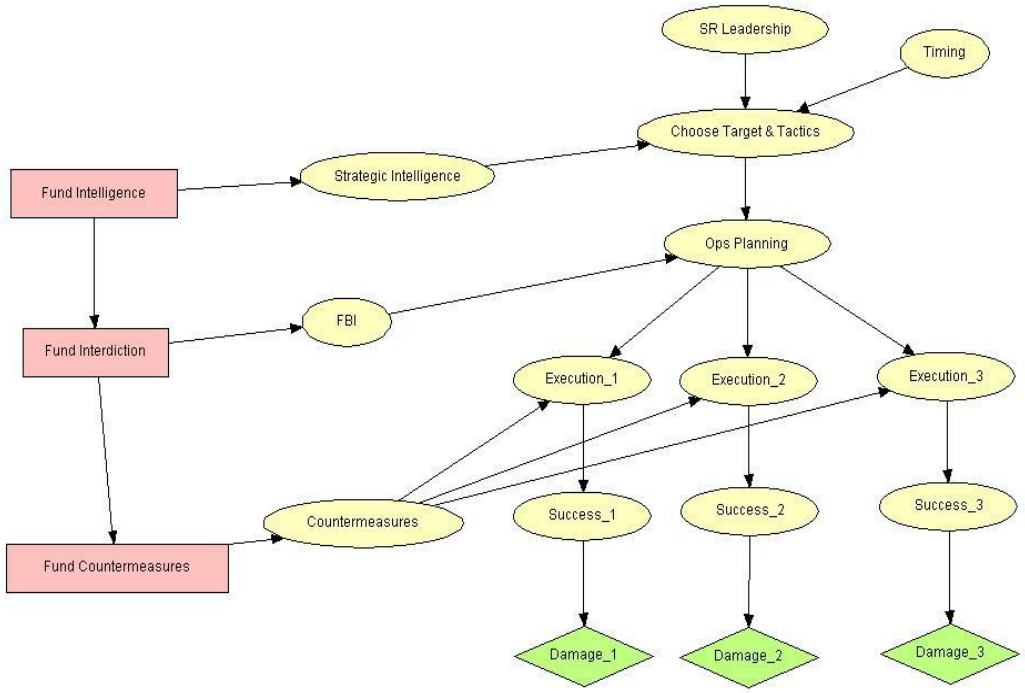

FIGURE 4 Counter-terrorist model for hierarchical threat organization

Multiple Models from Data

In recent years, the machine learning community has been moving to ensemble-based methods. Techniques such as bagging, boosting, and stacking have proven their value across a wide range of applications and constituent model types. From this, it can be inferred that ensembles of models often contain more information than any single model does, and a growing portfolio of means for exploiting ensembles of models are becoming available. I and colleagues have been using this approach to create ensembles of models from data. Instead of using model averaging as in common in machine learning, however, we use the resulting ensemble as a foundation for robust inference, as described above. In this section, I present two examples of this. In the first example, the ensemble consists of models with different structure whose parameters are estimated from the same data set. In the second example, the ensemble is created by bootstrap resampling of the training dataset, as is done in bagging.

The first example uses six regression models created by Paul Collier (Collier and Hoeffler 2000). Collier fit a wide variety of such models to a data set that contained historical data regarding internal conflict across 120 countries over the past forty years. Six of these models had high measures of goodness of fit. Though there is a maximally likely model, differences in likelihood across the six do not provide a statistical basis for rejecting any of the other five. These six models include four models whose predictors are consistent with a theory that conflict results from opportunity for overthrowing the government, aka “greed,” and one model whose predictors are consistent with the view that conflict results from “grievance.” The most likely model used combinations of these predictors.

In this exemplary analysis, these six models are used to forecast the probability of conflict in Pakistan in 2015. These forecasts involve assumptions regarding the continuation of observed trends and possible U.S. policy over the intervening years. By coupling these models with a policy effect model (based on expert elicitation), the impact of alternative U.S. policies on the probably of conflict in Pakistan can be assessed for each model.

This ensemble of models demonstrates that a strategy devised to be robust over all six models has better performance than the "optimal" strategy based upon the "best estimate" model. Figure 5 shows the structure and parameters of these six models.

Figure 6 displays the regret of the candidate strategy (the optimal strategy for the nominal case on the best estimate model) for random cases on all six models. The sampling technique used was a Latin hypercube (space-filling) design. As can be seen on from this graph, the candidate strategy has zero or small regret on all cases sampled from five of the models, and for some cases on the sixth. However, there are also some cases for the grievance model in which the regret is substantial, as much as a 60% increase in the probability of conflict.

Further analysis of the high regret cases can provide a great deal of insight. Figure 7 shows only the high regret cases from the previous figure, displaying for these cases the two most influential variables in producing them (based on an analysis of variance). This view allows us to say that the "optimal" strategy performs well unless it turns out that the grievance model is the right way to think about conflict in Pakistan. If the grievance model holds, there is a significant decline in democracy in the coming years, and the terrorist infrastructure is relatively proficient, then the optimal policy will perform badly compared with alternatives. If this strategy represented our best option, perhaps the decision maker would be willing to wager against this failure mode happening. But, as it turns out, further analysis can provide a better candidate and avoid the necessity of placing that bet.

ALL BETAS	Greed-NoPeaceDur	Greed-PeaceDur	Greed-GDP	Greed-Diasp	Grievance	Greed-Grievance (Base)
constant term	-15.56	-12.16	-3.704	0.7460	-1.688	-13.07
male schooling	-0.0336	-0.02470				-0.03156
ln GDP per capita			-0.8434	-1.237		
GDP growth-3*Pop growth	-0.1110	-0.1146	-0.0955			-0.1152
primary comm exports	19.91	18.93	16.58	17.57		18.94
primary comm exports^2	-31.79	-29.28	-23.42	-28.82		-29.44
ln population	0.8370	0.6775	0.4700	0.2949		0.7677
social fractionalization	-0.0001374	-0.0001590	-0.0002090		0.00005952	-0.0002135
geographic dispersion	-2.238	-2.115	-0.8242		0.2507	-2.487
mountainous terrain	0.01603	0.01326	0.007856		0.01144	
peace duration		-0.003636	-0.003678	-0.002010	-0.004566	-0.003713
diaspora/peace duration				700.9		
ethnic dominance					0.2361	0.6704
democracy					-0.1033	

FIGURE 5 Basis of work of World Bank's Paul Collier using data for several dozen conflicts for 120+ countries over 40 years

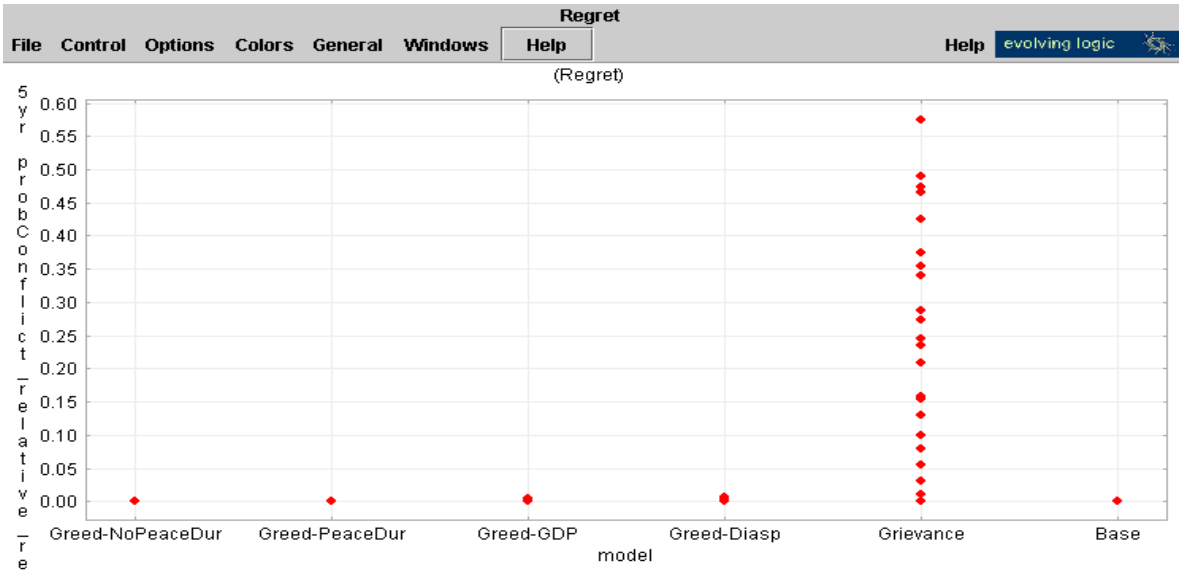

FIGURE 6 Optimal policy for best estimate model, which is vulnerable in the grievance model

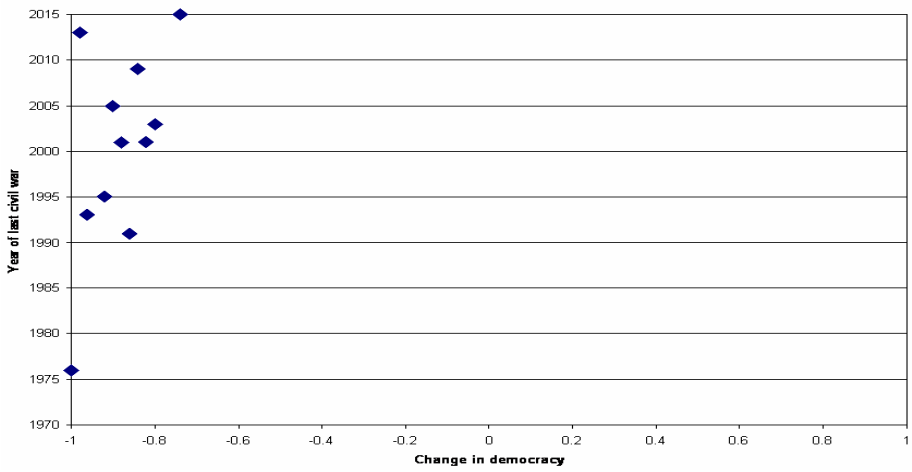

FIGURE 7 Characterizing the cases where optimal policy is vulnerable

A search over strategies reveals one that has the best overall performance on the high regret cases, as shown in Figure 8. Note that this strategy is not optimal for the nominal case for any of the models. Repeating the vulnerability analysis done for the best estimate strategy, the regret of this new strategy is calculated across a series of cases on each of the six models. Those results are displayed in Figure 8. This new strategy has non-zero regret for all of the models. But its maximum regret is two orders of magnitude less than the maximum for the best estimate strategy. (Note that the scale of Figure 8 is expanded compared to that of Figure 6.)

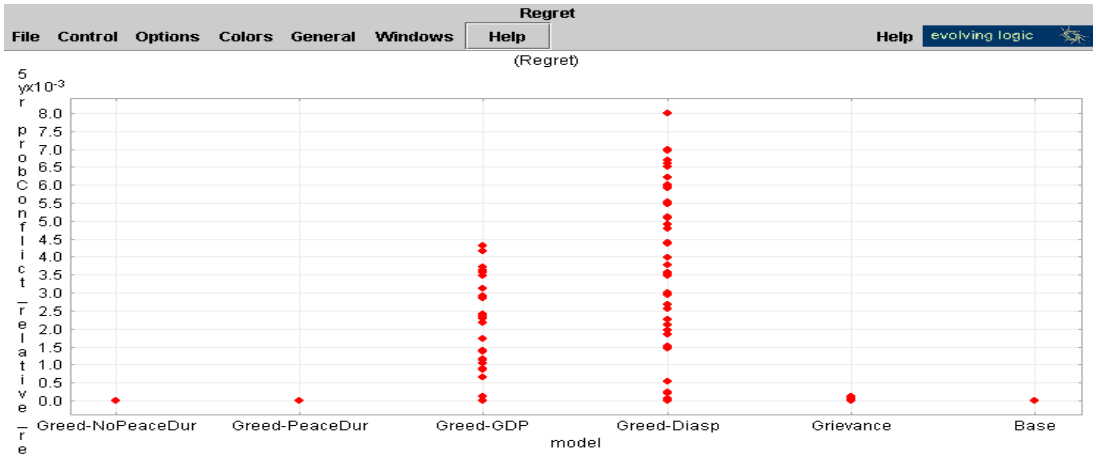

FIGURE 8 Vulnerabilities of candidate robust policy

The worst case for the hedging strategy has a relative regret of less than 1%. While further searching might reveal an even more robust strategy, this one is clearly much more robust than the original candidate, which was optimal for the best estimate model but had failure modes revealed by the ensemble of plausible models.

Had we used only the best model as the basis for our analysis, we would have recommended a strategy that was more vulnerable to surprise than the hedging strategy. The discovery of the hedging strategy was made possible by our use of the suite of alternative models. These different views, all derived from the same data, provide us with more information than any single model would have provided. If we are interested in robust strategies, the use of multiple alternative models provides a clear advantage.

This basic approach can be used with a wide variety of modeling structures. Figure 9, displaying the second example, is taken from a study in which an ensemble of feed-forward neural nets was created by bootstrap resampling among the training exemplars. This figure demonstrates that for new test instances, a variety of possibilities can be observed, from all of the members in the ensemble agreeing on the classification of this input to a broad distribution of classifications being observed across the network. Such an ensemble can be used as a challenge set to help us craft robust strategies.

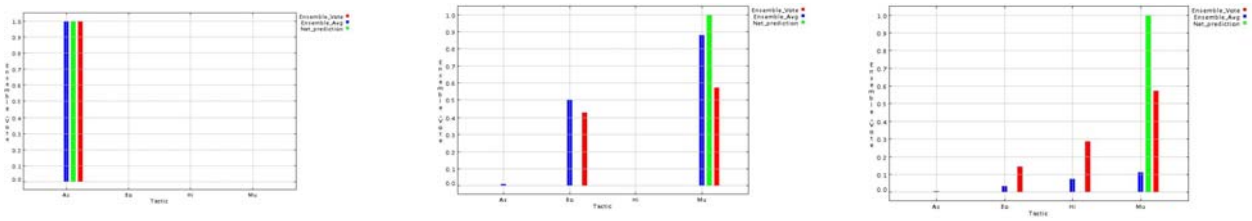

FIGURE 9 Predictions of model ensemble for a selection of novel test cases

Inference with Complementary Models

In addition to reasoning across ensembles of alternative models, computational science could benefit from an established method for joint inference across complementary models. Many efforts have been made historically to combine models by piping the output of one into the input of another, perhaps with some transformation of coordinates, such as miles to kilometers. However, such efforts have often been much less successful than hoped. This is because all models are approximations, and so are based on various assumptions. When the models used have unstated assumptions that contradict each other, essentially any conclusion can potentially result when they are combined. (Recall that from A and not-A, you can deduce anything.) Thus, combining models is in itself an act of modeling, requiring that assumptions of the combining be explicitly addressed and rectified against the pieces.

Treating models as representing ensembles of possible computational experiments can provide a much better basis for approaching the problem of model fusion than is possible through "wiring the models together." An environment for reasoning from modeling experiments (such as CARs) allows the "wiring" between the models to be set at "run time," and hence varied in response to changing hypotheses and goals. This allows strategies for specifying groups of computational experiments to account for uncertainties inherent in the model fusion itself. This approach allows the problem of model fusion to be transformed into a version of the machine learning problem.

To demonstrate this, the six Collier models are combined with two additional models of different type. The first is CAST, which digests daily news reports, and combines them using an expert-designed weighting scheme to produce 12 indicators of potential conflict. The second is a model based on structured interviews with military officers and government officials. This model represented the impacts differing strategies might have on various aspects of the future situation in Pakistan, along with their uncertainty about the size of these effects.

The inputs and outputs of these various models are not unrelated. In fact, the lever inputs to the Collier models and to the Effects model (Expert Elicitation model) are identical by design. The 12 indicators that are outputs of CAST can be related to various inputs to the Collier models. And outputs of the Collier models, particularly the probability of conflict by year, serve as an input to the Effects model.

CARs allows us to avoid expressing this relationships between the models as "hard wiring" created by revising the software. Such hard wiring often results in difficulties when models of aspects of a deeply uncertain problem are combined in this way. The scheme for joining two models in this way is itself an act of modeling, and, as with other models of deeply uncertain phenomena, the assumptions embodied in the wiring itself represent an uncertainty whose implications should be explored. Failure to contend with this difficulty has at time led to nonsensical results in past efforts at model fusion.

Instead of hard wiring, we continue to treat each model as a separate platform for computational experiments, but allow the strategy for case generation for experiments on various platforms to be coupled or correlated in arbitrary ways. This allows us to treat the constellation of models as a single unified model if we wish to, but also allows the confederation of models to be exercised in other ways, to answer other questions than "what would happen if."

REFERENCES

Bankes, S., 1993, "Exploratory Modeling for Policy Analysis," *Operations Research*, Vol. 41, No. 3, pp. 435–449.

Bankes, S., and R. J. Lempert, 2004, "Robust Reasoning with Agent Based Models," *Nonlinear Dynamics, Psychology, and Life Sciences*.

Boehm, B., 1988, "A Spiral Model of Software Development and Enhancement," *IEEE Computer*.

Boehm, B., 2000, *Spiral Development: Experience, Principles, and Refinements*, Special Report CMU/SEI-00-SR-08, ESC-SR-00-08, Software Engineering Institute, Carnegie Mellon University; available at www.sei.cmu.edu/cbs/spiral2000/february2000/BoehmSR.html.

Collier, P., and A. Hoeffler, 2000, "Greed and Grievance in Civil War," World Bank Policy Research Working Paper No. 2355; available at http://ssrn.com/abstract=630727.